A CONSTRAINT SATISFACTION NEURAL NETWORK FOR CASE ANALYSIS

WEE KHENG LEOW
ROBERT F. SIMMONS

MARCH 1990    AI90-129
A CONSTRAINT SATISFACTION NEURAL NETWORK
FOR CASE ANALYSIS†

Wee Kheng Leow
Robert F. Simmons

Department of Computer Sciences
and
Artificial Intelligence Laboratory
Taylor Hall 2.124, University of Texas at Austin
Austin, Texas 78712-1188

leow@cs.utexas.edu
simmons@cs.utexas.edu

Abstract

In this paper, we describe a neural network which performs case analysis using constraint satisfaction. Its design is based on a careful analysis of the computational requirements and constraints posed by the problem. This network is superior to those published in that it learns to capture rules similar to those used in symbolic systems. Consequently, it requires only minimal training to achieve good performance. Furthermore, its performance will not degrade as rapidly as the other networks when the problem size scales up.

Keywords:

Case analysis, constraint satisfaction, generalization, Hopfield net, Mean Field Theory, position invariance of influence, rule-based processing, two-phase learning.

† Support for this project was provided in part by Microelectronics and Computer Technology Corporation, and by the U.S. Army Research Office under contract DAAG29-84-K-0060.
1. INTRODUCTION

Recently, there has been an increase in the interest in employing neural networks for solving natural language understanding problems (e.g. [CoSm83], [Dyer88], [McKa86], [SJMc88], [WaPo85]). Undoubtedly, neural networks have some advantages, as well as disadvantages, over classical symbolic systems ([Dyer88]). In our opinion, their greatest strength lies in their abilities to process information in massive parallelism, to handle erroneous inputs and perform best matches, and to learn to capture regularities, or rules, present in the problem environment. Their greatest weakness, on the other hand, is their disparity from high level reasoning processes. Their strength is precisely the weakness of symbolic systems, and vice versa.

Granted that one wishes to complement the weakness of symbolic systems by the strength of neural networks, how should one design networks that perform the required tasks? We find that many published works in this area are rather ad hoc, much like the initial stages of development of the symbolic paradigm. This immediately reminds us of the three-level approach proposed by Marr ([Marr82]). Although we may not conform to the approach strictly, we are in favor of employing the guidelines suggested by the approach – for one is unable to develop a suitable system unless one states clearly the requirements and constraints posed by the problem.

Before discussing our network and how we arrive at the design, we will first discuss some of the networks proposed by other researchers and their major weaknesses (Section 2). This lays a good foundation on which we carefully analyze the problem that we intend to solve, i.e., case analysis of English sentences (Section 3). Such analysis makes explicit the requirements and constraints posed by the problem which, in turn, specifies a natural way to design the network (Section 4). We conducted experiments with the network and obtained encouraging results (Section 5). We think that our network is superior to those published in that it learns to capture rules similar to those used in symbolic systems. Consequently, it requires only minimal training to achieve good performance. Furthermore, its performance will not degrade as rapidly as the other networks when the problem size scales up.

2. LITERATURE REVIEW

The design of neural networks for case analysis has been previously reported in [McKa86] and [SJMc88]. McClelland and Kawamoto ([McKa86]) employed a simple feed-forward network (Fig. 1) with one layer of adjustable weights trained by perceptron convergence procedure ([Rose62], [MiPa88]). The input units, which encode a canonical form of the input sentence, are divided into four groups, one for each word, in the order: verb, subject NP, object NP, and with-NP. The actual inputs to the input units are transformed, word by word, from the input words. The output units are also divided into four groups, one for each case-filler relationship, in the order: AGENT, PATIENT, INSTRUMENT, and MODIFIER. A reverse transformation is performed on these groups of output units to interpret the actual output words.

The input and output words are encoded using two sets of microfeatures, one for verbs (VF), and one for nouns (NF). Each group of input units is either a matrix of VF×VF (for verbs) or a matrix of NF×NF (for nouns). Similarly, each group of output units is a matrix of
NF×NF. In other words, each input unit, and likewise for output unit, corresponds to a conjunction of two microfeatures.

The objective is to train the network to fill the case roles with the correct input words. For example, with the canonical input sentence

\[
\text{broke boy window hammer}
\]

which means "The boy broke the window with the hammer", the expected output is

\[
\text{boy window hammer (nil)}
\]

i.e., \(\text{AGENT} = \text{boy}, \text{PATIENT} = \text{window}, \text{INSTRUMENT} = \text{hammer}, \text{and MODIFIER} = \text{(nil)}\).

This network has several major weaknesses:

1. The many-to-one transformation (\(\text{NF} \times \text{NF} \rightarrow \text{NF}\)) from output matrices of microfeatures to output words (vectors of microfeatures) is hand coded and is, thus, extremely difficult to optimize.

2. Significant amount of preprocessing, including parsing and verb-noun disambiguation, is required to transform raw sentences into canonical forms before this network can work on them. The capacity of the network to perform several distinct processes in parallel, thereby effectively using contextual constraints ([WaPo85]), has been omitted.

3. The network lacks an important quality which we call \textbf{Position Invariance of Influence} of words (refer to Section 3 for detail discussions). For example, if the words "boy" and "girl" have only appeared respectively as \text{AGENT} and \text{PATIENT} in the training sentences, then the network would not respond correctly to sentences in which "girl" is the \text{AGENT} and "boy" is the \text{PATIENT}. Although the authors have encoded the words such that words sharing common features will have similar input patterns, we argue that such encoding is, in itself, insufficient to ensure Position Invariance of Influence. The reason is that the network is basically performing pattern classification, and it can, and will, pick up any peculiarity (also called noise) in the training patterns that helps it classify the training patterns. For example, referring to the previous example, the network will associate the features of "boy" with \text{AGENT}, and the features of "girl" with \text{PATIENT}. Unless "boy" acts as \text{AGENT} in some sentences and as \text{PATIENT} in the others, the network will not be trained correctly. However, if one really includes all such sentences, then one has to do likewise for numerous other words such as "Jim", "May", "man", "woman", "he", "she", etc. Without a doubt, one will run into combinatorial explosions on the number of input sentences.

Now, let us examine St. John and McClelland's network ([SJMc88]). Their network was a 5-layer recurrent network (Fig. 2) that was designed to perform case analysis and other natural language tasks. Here, we will only discuss the case analysis aspect. In their network,
Fig. 1. McClelland and Kawamoto's network ([McKa86]).

Fig. 2. St. John and McClelland's network ([SJMc88]).
input units are divided into three groups, namely position, phrase, and sentence gestalt. A phrase is simply a verb or a noun along with any adverb or preposition. Within the phrase group, a unit is used to represent each verb, noun, adverb, and preposition. The position group encodes the positions of the phrases with respect to the verb. Four position indicators are used, each represented by one unit, namely, verbal, pre-verbal, 1st-post-verbal, and n-post-verbal. Position indicators are needed because only one phrase is input to the network at a time (the advantage of doing so is that the network could potentially handle sentences of indefinite lengths). For example, the sentence

The ball was hit by someone with the bat in the park

is encoded and input in the order

pre-verbal/ball
verbal/(was, hit)
1st-post-verbal/(by, someone)
n-post-verbal/(with, bat)
n-post-verbal/(in, park)

The output units are divided into two groups, one for the case roles, and the other for the words that fill the case roles. One unit is used to represent each of the case roles, nouns, actions, adverbs, and additional features (such as male and female).

During training, a sentence gestalt (a kind of internal representation) is formed based on the current input and the previous sentence gestalt. The current sentence gestalt is then copied to form part of the input for the next input cycle. After the whole sentence has been processed, if the network is probed at the probe units with the input words, then appropriate role units should turn on, and the other role units should turn off.

The major weaknesses of this network are:

1. The authors reported that when infrequent or irregular sentences were processed, the information carried by the inputs may be erroneously overridden by learned constraints. This indicates that the network could not capture rules properly, since rules that are properly captured will apply to any sentence regardless of whether they are frequently or infrequently seen (see Section 5 for further discussions).

2. The authors reported that the network had the tendency to activate filler units for case roles that do not apply to a particular case frame. This again suggests that the network could not capture rules properly.

3. This network also suffers from the drawback that it does not embody Position Invariance of Influence. A word (or phrase) is tagged as either pre-verbal, verbal, 1st-post-verbal or n-post-verbal. If a noun, say "boy", has only appeared in training sentences as, say, pre-verbal, then, the network would not respond correctly to sentences with "boy" appearing as post-verbal. Encoding similar words with similar representations
Neural Network for Case Analysis

does not help, as for the previous network ([McKa86]), since this network is basically performing pattern classification.

Besides observing these major weaknesses, we also find it hard to compare the performances of the networks. The authors did not provide clear indications of performance. In [McKa86], the authors reported that about 85% of the microfeatures that should be on were turned on, and about 0.6% of the microfeatures that should be off were also turned on (averaging over familiar and novel sentences). In [SJMc88], no assessment of generalization was made.

3. ANALYSIS OF THE PROBLEM

In this section, we will analyze the problem of case analysis carefully, and identify the requirements and constraints posed by the problem. We will first look at issues pertaining to case analysis, and then follow with more general issues.

3.1. Case Grammar Theory

It is well understood that case analysis is a verb-centered process governed by rules that the Linguistics and AI communities have studied for quite some time ([Alle87], [Cook89]). A noun or noun phrase (NP) may fill different case roles in different sentences, but the exact role to fill in a particular sentence is fundamentally decided by the main verb of the sentence ([Cook89]). Different verbs expect different sets of case roles (also called case frames). The verb "break" has a case frame consisting of OBJECT, and optional AGENT and INSTRUMENT cases; whereas the verb "give" expects AGENT, OBJECT and RECIPIENT cases. The assignment of cases to NPs is also governed by rules such as word order, syntactic and semantic constraints, etc ([Alle87], [Cook89], [McKa86], [SJMc88]). This suggests that case analysis should be modeled as a constraint satisfaction problem so that rules can be captured as constraints.

In Fillmore's 1968 model of case grammar ([Fill68]), the deep structure of a sentence (S) consists of two parts: a modality (M), and a proposition (P). The proposition (P) consists of a central verb (V) and a series of case-marked noun phrases (C). Each case-marked noun phrase (C) consists of a case marker (K) and a noun phrase (NP). Putting it more succinctly, the deep structure can be generated by the rules:

\[
S \rightarrow M \ P \\
P \rightarrow V \ C \ C \ldots \ C \\
C \rightarrow K \ NP
\]

For example, the sentence

John broke the window with a hammer

has the deep structure shown in Fig. 3. The NP that fills the OBJECT case may have no case marker, as in this example.
Fillmore also made some assumptions about case frames. The most important one is that no case role may appear more than once in a (simple) sentence. Apparent counterexamples are to be explained as two different cases or as examples of complex sentences.

Cook ([Cook89]) proposed that all the verbs can be categorized into twelve types along two dimensions. Along one of these dimensions, verbs are categorized as either Basic, Experiencer, Benefactive or Locative. Along the other dimension, verbs are categorized as either State, Process or Action. Examples of each type of verb are shown in Table 1. Some verbs may belong to more than one verb type when used differently. For example, the verb "break" is a Basic Process verb when used as an intransitive verb, and a Basic Action verb when used as a transitive verb.

By ignoring modality and making minor simplifications, Cook showed that each verb type only has a small number of case frames shared by all the verbs belonging to the same verb type. The verbs belonging to the same verb type can be further divided into smaller groups. Table 2 shows the basic case frames associated with a verb type. The INSTRUMENT case is not specified in the table since it can be associated with almost all the verb types. Similarly, for TIME case.

The BENEFACTIVE case in Cook's formulation is more general than the one usually used in Natural Language Processing ([Alle89]). It includes the owner and receiver of objects. Benefaction may be either positive or negative, and the benefactor may be a gainer or a loser. Similarly, for the LOCATIVE case in Table 2. In practice, one may expand BENEFACTIVE and LOCATIVE cases as suggested in [Allen89], but the general idea of a case frame matrix, such as Table 2, is still useful.

Cook also pointed out the importance of Covert Case Roles. He argued that without covert case roles, it would be impossible to classify each verb according to the full

<table>
<thead>
<tr>
<th>verb types</th>
<th>Basic</th>
<th>Experiencer</th>
<th>Benefactive</th>
<th>Locative</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>be tall</td>
<td>like</td>
<td>need</td>
<td>sit</td>
</tr>
<tr>
<td>Process</td>
<td>break (iv)</td>
<td>remember</td>
<td>find</td>
<td>sink</td>
</tr>
<tr>
<td>Action</td>
<td>break (tv)</td>
<td>taste</td>
<td>give</td>
<td>put</td>
</tr>
</tbody>
</table>

Table 2. Basic case frame matrix. A = AGENT, E = EXPERIENCER, B = BENEFACTIVE, L = LOCATIVE ([Cook89]).

<table>
<thead>
<tr>
<th>verb types</th>
<th>Basic</th>
<th>Experiencer</th>
<th>Benefactive</th>
<th>Locative</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>O</td>
<td>E, O</td>
<td>B, O</td>
<td>O, L</td>
</tr>
<tr>
<td>Process</td>
<td>O</td>
<td>E, O</td>
<td>B, O</td>
<td>O, L</td>
</tr>
<tr>
<td>Action</td>
<td>A, O</td>
<td>A, E, O</td>
<td>A, B, O</td>
<td>A, O, L</td>
</tr>
</tbody>
</table>
complement of case roles that the verb demands. Covert case roles are roles that are present in the deep structure, but are sometimes or always absent from the surface structure. Covert case roles may be partially or totally covert. Partially covert case roles are sometimes present and sometimes absent, and are also known as Deletable case roles. For example, in the following sentences, the NP’s in the brackets may be present or absent. Their corresponding case roles are deletable, denoted by "-del".

Mary /told /me /(something).
A V E O-del.

May /said /something /(to me).
A V O E-del.

Totally covert roles are roles that are always absent in the surface structure, and they include Coreferential roles and Lexicalized roles. Coreferential roles are case roles that are applied to a single noun. For example,

John /moved /to Austin.
A, O V L

Mary /borrowed /the book.
A, B V O

Lexicalized roles are case roles that are incorporated into the surface verb form. So, they do not appear in the surface structure. For instance,

He /watered /the plant.
(the OBJECT water is lexicalized)

The glass /sank.
(the LOCATION to which the glass sinks is lexicalized)

3.2. General Issues

In this section, we discuss general issues that may pertain to other Natural Language Processing tasks as well.

As we know, natural languages are full of ambiguous words and sentences. How does human learn to understand the meanings of words and sentences? We postulate that at least two modes of learning are involved. In the word-learning mode, we learn the possible meanings of words, and in the sentence-learning mode, we learn to constrain the possible meanings in the context of the sentence. Two-Mode Learning seems especially evident in students learning foreign languages. It helps students learn to construct sentences. For example, after learning how to construct the sentence

John ate the chicken,
Fig. 3. Deep structure for "John broke the window with a hammer" (adapted from [Fill68]).

Fig. 4. An IR cluster. Most of the connections are mutually inhibitory.

Fig. 5. Word-IR subnetwork. The units in the IR cluster are mutually unconnected.
and knowing that beef can also be consumed, a normal person seems to have little difficulty constructing a similar sentence

    John ate the beef.

In fact, a foreign language student may even construct sentences such as

    John ate the water.

Although the last sentence is inappropriate in English†, it has some semantic validity in that water, like chicken and beef, can be consumed.

The next issue we like to raise has to do with the training of networks. Consider a multilayer feedforward network trained to preformed case role assignment. Suppose that the network has been trained with sentences in which the word "John" appears only as AGENT, and the word "Mary" appears only as "RECIPIENT", as in

    John gave Mary the book.

Then, the network will not respond correctly to sentences with "John" and "Mary" playing the opposite roles, as in

    Mary gave John the book.

The reason is that the weights have not been trained to associate "John" with RECIPIENT and "Mary" with AGENT. Ideally, since "John" and "Mary" share the common property of being able to fill the AGENT and RECIPIENT roles, the network should be able to deduce the correct assignment for the second sentence from that for the first sentence. We call this property Position Invariance of Influence of the words.‡ In other words, independent of the positions of the words "John" and "Mary", both should have the capacity to fill the AGENT and RECIPIENT roles. Furthermore, after learning that "John" is the AGENT and "Mary" is the RECIPIENT in some sentences, the network should be able to generalize to similar sentences in which "John" and "Mary" fill opposite roles.

Usually, what one would do to capture Position Invariance of Influence is to encode words sharing common features with similar representations, such as distributed representation ([HMR86]). However, we argue that such encoding is, in itself, insufficient to ensure Position Invariance of Influence. The reason is as follows. Referring to the previous example, since "John" and "Mary" are associated with different case roles during learning, the network will adjust its weights to pick up dissimilarities between the representations of "John" and "Mary" to distinguish them. Similarity in their representations will only be regarded as redundant since it does not provide any information for distinguishing "John" and "Mary".

A naive way to counter this problem is to provide training sentences in which both "John" and "Mary" appear as AGENT and RECIPIENT with roughly equal probabilities.

† Incidentally, this sentence is appropriate in some dialects of Mandarin.
‡ The word "Influence" is used in the context of constraint satisfaction.
Unfortunately, this method is impractical since one would have to do likewise for numerous other words such as "Jim", "May", "man", "woman", "he", "she", etc. Moreover, one has to include enough training sentences to ensure that every noun fills every possible case role in every possible combination of words making up the sentences. Undoubtedly, one will soon run into combinatorial explosions on the number of input sentences.

3.3. Assumptions, Requirements and Constraints

After analyzing the problem, we can now state our assumptions, and summarize the requirements and constraints posed by the problem.

Assumptions
1. We consider only simple sentences in active voice with only one verb. Modality such as tense, aspect and mood are omitted.

2. We also omit determiners. Therefore, only three types of words are considered in a sentence, namely verbs, nouns and case markers (i.e. prepositions).

3. Following Fillmore’s model, we assume that no case role may appear more than once in a sentence.

4. Each noun fills exactly one case role. This assumption may be relaxed to handle covert case roles.

Requirements and Constraints
1. Case analysis is a verb-centered process.

2. Case analysis should be modeled as constraint satisfaction so that the network can capture the rules present in the problem environment effectively and efficiently.

3. The network should incorporate the Two-Mode Learning postulate.

4. The network should embody the Position Invariance of Influence.

4. DESIGN OF NETWORK

The fundamental design of our network rests on the premise of Two-Mode Learning postulate. Although these two modes are necessarily integrated, as a form of simplification, we will separate them into two separate phases. The first phase is the learning of possible case roles that a noun can fill, and the second phase is the learning of the actual case role that a noun fills in the context of the sentence. We shall call the collection of possible case roles that a noun can fill the Intermediate Representation (IR) of the noun. Since a word may be a noun or a verb in different context (e.g. break), IR must represent the verb types as well. For clarity of exposition, we will refer to the group of input units that receive a word as the word
cluster, and the group of IR units that represent the IR of a word as the IR cluster. So, a word cluster and an IR cluster is associated to each word in a sentence.

When a word is presented to the network, the word cluster must activate the correct IR units to represent the IR of the word. Since a word may be input at any position within the sentence, a word cluster may receive a noun, a verb or a case marker. To achieve Position Invariance of Influence, the IR cluster must have the provision to represent not only different verb types and case roles, but also different case markers. Thus, one is forced to construct IR clusters that combine a verb type cluster, a case cluster and a case marker cluster (Fig. 4). We designate a unit to represent each verb type, case and case marker in an IR cluster. By Assumptions 2 and 4, the connections within the IR cluster should be mutually inhibitory (see following sections for exceptions).

The transformation of words into IR’s can be achieved by an ordinary feedforward network. We have chosen a network with one hidden layer (Fig. 5) so that the input patterns need not be linearly-separable. The weights of this network can be trained using either back-propagation (BP) learning algorithm ([RHW86]), Boltzmann machine (BM) learning algorithm ([AHS85]) or Mean Field Theory (MFT) learning algorithm ([PeAn87]).† We have arbitrarily chosen MFT just to find out how well it performs in this task.

The word-IR subnetwork is replicated as many times as there are words in a sentence, in this, seven. This kind of architecture places a limit on the length of the sentences. At the moment, there is no proven good method to counter this problem (see Section 5.3.4 for our suggestions). The weights in the feedforward parts of all the word-IR subnetworks are identical. Independent of the position of a word within the sentence, the subnetworks will always produce the same IRs in the IR clusters. The IR units that are finally turned on or off are decided by constraint satisfaction operating only on the IR units. This is the key design in achieving Position Invariance of Influence.

The second learning phase involves the learning of the actual IR units to be turned on or off in the context of the entire sentence. As we have discussed earlier, this process should be modeled as constraint satisfaction. One network that can perform constraint satisfaction well is the Hopfield net ([HoTa85], [AHS85], [PeAn88]). So all the IR clusters must be connected to form a Hopfield net. Instead of connecting the IR clusters that take inputs from the hidden layer, we construct duplicates of the IR clusters and feed the outputs of the first IR layer, units for units, to the duplicates (Fig. 6). The units in the first IR layer are mutually unconnected as in feedforward network. The units in the second IR layer are connected to form a Hopfield net (with symmetric connection weights). In this way, inputs to the second IR layer are maintained by the first IR layer while constraint satisfaction is operating on the second IR layer. The final outputs of the second IR layer (which also serves as the output layer) thus represent the results that best match the learned constraints. For example, given the input sentence

John gave book to Mary

the output at the second IR layer will be

† Theoretically, BM and MFT assume symmetric connection weights. So, the network should be modified
where BA abbreviates Benefactive Action verb, and MR abbreviates case marker for RECIPIENT. Outputs of IR units corresponding to null input words will have no activation (null IRs).

The weights of the connections among the 2nd-layer IR units are trained using MFT learning algorithm (since MFT algorithm runs faster than BM algorithm in general ([PeAn88], [PeHa89])). These weights can be initialized appropriately to speed up the learning process. These initial values should be consistent with the final values, i.e., inhibitory connections should be initialized with negative random weights and excitatory connections should be initialized with positive random weights. From the assumptions, requirements and constraints, we obtain the following initial settings:

1. All the units within an IR cluster are mutually inhibitory (Assumptions 2 and 4).

2. Each verb type unit inhibits all other verb type units (Assumption 1).

3. Each verb type unit excites or inhibits case units and case marker units in all other clusters (Requirement 1). So, the corresponding connection weights may be randomly initialized.

4. Each case unit inhibits all other case units of the same type (Assumption 3).

5. Each case marker unit excites case units of the same type in the next cluster, and inhibits case units of different types in the next cluster (since a case marker suggests the role of the noun following it).

6. Each case marker unit inhibits all case marker units in the next cluster (since the word following a case marker cannot be a case marker).

7. Each case marker unit inhibits all other case marker units of the same type (Assumption 3).

8. Any other connection should have fixed zero weight, i.e. not connected.

To handle covert case roles, Assumption 4 must be relaxed. This does not pose any difficulty to our network since it only enforces soft constraints. As opposed to hard constraints, which are constraints that must be satisfied, soft constraints are constraints that should be satisfied as much as possible. In this respect, neural networks are more tolerant to violation of constraints and will continue to process in these situations. On the contrary, symbolic systems will either breakdown, or need to have their rules modified which can be a painful appropriately for best learning results.
task.

Our network can handle coreferential and lexicalized case roles but not deletable case roles. For coreferential roles, the IR cluster corresponding to the noun with two roles will have both the case units turned on. For lexicalized roles, the IR cluster corresponding to the verb will have the appropriate case unit turned on. We have omitted deletable roles because they entail the activations of case units in IR clusters that do not correspond to any input word. In practice, deletable case roles are usually omitted since one is interested in assigning case roles to the input words rather than finding roles that are unfilled ([Alle89]).

5. EXPERIMENTS AND RESULTS

5.1. Network Specifications

In our experiments, we constructed a network with the following specifications:

5.1.1. Verb Type Cluster

12 verb type units are allocated, one for each verb type, as in Table 3.

5.1.2. Case Cluster

One unit is allocated to each case. We expanded the BENEFACTIVE case of Cook ([Cook89]) to include OWNER, FROM-POSSESSOR, and RECIPIENT. We also expanded LOCATIVE case to include SOURCE location and DESTINATION location ([Alle87]). The list of 11 cases considered are given in Table 4.

5.1.3. Case Marker Units

One unit is allocated to each case marker. Table 5 gives the list of 7 case markers considered.

5.1.4. Input Word Encoding

To make the learning of word-to-IR mapping easier, we employed a type-instance encoding method (Fig. 7). This type of encoding has the advantage that words sharing common properties have the same type codes.

Table 3. Verb Type Units.

<table>
<thead>
<tr>
<th>verb types</th>
<th>Basic</th>
<th>Experiencer</th>
<th>Benefactive</th>
<th>Locative</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>SS</td>
<td>ES</td>
<td>BS</td>
<td>LS</td>
</tr>
<tr>
<td>Process</td>
<td>PP</td>
<td>EP</td>
<td>BP</td>
<td>LP</td>
</tr>
<tr>
<td>Action</td>
<td>AA</td>
<td>EA</td>
<td>BA</td>
<td>LA</td>
</tr>
</tbody>
</table>
Fig. 6. The network for case analysis. Total 7 subnetworks, $80 \times 7 = 560$ units. The units in the 1st IR layer are mutually unconnected.

null word

verb or noun

<table>
<thead>
<tr>
<th>verb type</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>111</td>
</tr>
<tr>
<td>experiential</td>
<td>100</td>
</tr>
<tr>
<td>benefactive</td>
<td>010</td>
</tr>
<tr>
<td>locative</td>
<td>001</td>
</tr>
<tr>
<td>non-verb</td>
<td>000</td>
</tr>
</tbody>
</table>

noun type

code
e.g.

<table>
<thead>
<tr>
<th>noun type</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>animate</td>
<td>10</td>
</tr>
<tr>
<td>physical inanimate</td>
<td>01</td>
</tr>
<tr>
<td>non-physical</td>
<td>11</td>
</tr>
<tr>
<td>non-noun</td>
<td>00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>case marker type</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>locative</td>
<td>10</td>
</tr>
<tr>
<td>non-locative</td>
<td>01</td>
</tr>
<tr>
<td>non-case marker</td>
<td>00</td>
</tr>
</tbody>
</table>

Fig. 7. Encodings of input words.
Table 4. Case Units.

<table>
<thead>
<tr>
<th>case</th>
<th>abbr</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENT</td>
<td>A</td>
</tr>
<tr>
<td>OBJECT</td>
<td>O</td>
</tr>
<tr>
<td>EXPERIENCER</td>
<td>E</td>
</tr>
<tr>
<td>BENEFICIARY</td>
<td>B</td>
</tr>
<tr>
<td>INSTRUMENT</td>
<td>I</td>
</tr>
<tr>
<td>OWNER</td>
<td>P</td>
</tr>
<tr>
<td>FROM-POSSESSOR</td>
<td>F</td>
</tr>
<tr>
<td>RECIPIENT</td>
<td>R</td>
</tr>
<tr>
<td>LOCATION</td>
<td>L</td>
</tr>
<tr>
<td>SOURCE</td>
<td>S</td>
</tr>
<tr>
<td>DESTINATION</td>
<td>D</td>
</tr>
</tbody>
</table>

Table 5. Case Marker Units.

<table>
<thead>
<tr>
<th>case marker for</th>
<th>abbr</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT</td>
<td>MO</td>
</tr>
<tr>
<td>INSTRUMENT</td>
<td>MI</td>
</tr>
<tr>
<td>FROM-POSSESSOR</td>
<td>MF</td>
</tr>
<tr>
<td>RECIPIENT</td>
<td>MR</td>
</tr>
<tr>
<td>LOCATION</td>
<td>ML</td>
</tr>
<tr>
<td>SOURCE</td>
<td>MS</td>
</tr>
<tr>
<td>DESTINATION</td>
<td>MD</td>
</tr>
</tbody>
</table>

We chose a total of 30 verbs, 27 nouns and 8 case markers. The complete list is given in the appendix. There are four ambiguous words, each can be a noun or a verb. These words are listed in Table 6. For these words, both the verb-type parts and the noun-type parts of their codes are set accordingly. There are altogether 61 words. A sentence generating

Table 6. Ambiguous words.

<table>
<thead>
<tr>
<th>verb</th>
<th>when used as noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>as in &quot;tea break&quot;</td>
</tr>
<tr>
<td>show</td>
<td>movie, performance</td>
</tr>
<tr>
<td>fall</td>
<td>water-fall, Autumn</td>
</tr>
<tr>
<td>sink</td>
<td>washing basin</td>
</tr>
</tbody>
</table>
program generates up to 12000 sentences for training and testing. Not all possible usage of the words are included.

5.2. Experimental Results

The 1st learning phase involved the learning of the mapping from input word to IR. Only one word-IR subnetwork was trained. After training, the weights were copied to all other word-IR subnetworks. The MFT algorithm was used to train the subnetwork with learning rate of 0.05, and annealing schedule of $1@30\times0.7^t$, $0 \leq t \leq 10$. All 61 word-IR mappings were correctly learned after running for about 300 epochs. Each epoch involved the presentation of the complete set of 61 word-IR pairs. The actual IR outputs were all within 10% error margin of the desired outputs. Most of them were, in fact, practically 1 or 0 as required.

The 2nd phase involved the learning of the actual case roles in the context of the entire sentence. Since many words have the same IR’s, the 12000 input sentences could be mapped into just 293 IR sentences, about 1/40 of the total number of input sentences. Instead of feeding input sentences to the input layer, we fed IR sentences directly to the 1st IR layer. Thus, we reduced the amount of input-output pairs from 12000 to at most 293. During training, the correctness criterion was set at 0.2, i.e., if the actual output was within 20% error margin of the target output, it was considered as correct. During testing, the correctness criterion was set at 0.5. Learning rate and annealing schedule were the same as those in Phase 1.

In order to investigate the performance of our network in term of training set size, we trained the network independently with two different training sets, A and B. 46 IR sentences were chosen for Set A and 145 IR sentences were chosen for Set B, representing respectively 15.7% and 49.5% of the total 293 IR sentences. Our choice of the sentences was quite arbitrary. We only ensured that every verb participated in some of the chosen sentences.

With Set A (46 IR sentences), the network learned to process 43 IR sentences correctly. But, amazingly, it generalized correctly to 171 novel IR sentences, i.e., a generalization capability of 3.7 times the training set size. These 214 (= 171 + 43) correctly processed IR sentences could account for about 73.0% (= 214 / 293) of the input sentences (about 8800 input sentences).† For each of the IR sentences that were incorrectly processed, the network erred in only about 2 output IR units among 210 of them. The training took only about 50 epochs to reach optimal performance level after which further training did not improve the performance significantly.

With set B (145 IR sentences), the network learned to process 137 IR sentences correctly. After training, it could generalize to 120 novel IR sentences. These 257 (= 137 + 120) correctly processed IR sentences could account for about 87.7% (= 257 / 293) of the input sentences (about 10500 input sentences). For each of the IR sentences that were incorrectly processed, the network again erred in only about 2 units among 210 units. The training took only about 50 epochs to reach optimal performance level after which the performance declined slowly with further training. The reason for this decline in performance is not clear. Table 7 and 8 give a summary of the performance of the network.

† Actually, some IR sentences correspond to more input sentences than the others. For simplicity, we assume that the distribution is roughly uniform.
Table 7. Summary of performance (in terms of sentences).

<table>
<thead>
<tr>
<th></th>
<th>training set size (%total†)</th>
<th>training performance</th>
<th>testing performance‡‡</th>
<th>overall performance*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size</td>
<td>performance</td>
<td>performance</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>46 (15.7%)</td>
<td>43 (93.5%)</td>
<td>214 (73.0%)</td>
<td>8800 (73.0%)</td>
</tr>
<tr>
<td>B</td>
<td>145 (49.5%)</td>
<td>137 (94.5%)</td>
<td>257 (87.7%)</td>
<td>10500 (87.7%)</td>
</tr>
</tbody>
</table>

† Total number of IR sentences = 293.
‡ Include training sentences.
* Total number of input sentences = 12000.

Table 8. Summary of performance
(in terms of bit-error, i.e. output IR units erroneously turned on or off).

<table>
<thead>
<tr>
<th>training set</th>
<th>training performance</th>
<th>testing performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size</td>
<td>bit-error†</td>
</tr>
<tr>
<td>A</td>
<td>46</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>145</td>
<td>18</td>
</tr>
</tbody>
</table>

† Total number of bit-error.
‡‡ Number of bit-error averaged over IR sentences incorrectly processed.
* Number of bit-error averaged over all IR sentences processed.
Numbers in brackets are percentage bit-errors. Total number of output IR units = 210.

5.3. Analysis of Performance

5.3.1. Learning Deficiencies

For both the learning trials, the network could not learn up to 100% accuracy. The main reason is that there is inconsistency in the output requirements. For the first learning trial (Set A, 46 IR sentences), two of the three incorrectly processed IR sentences have inconsistent output requirements: There is an OBJECT in each sentence, and the verb types are both BS. But, the subject NPs are required to fill different case roles.

<table>
<thead>
<tr>
<th>input sentence</th>
<th>active 1st-layer IR units</th>
<th>desired output</th>
<th>actual output</th>
</tr>
</thead>
<tbody>
<tr>
<td>owned (water-)fall</td>
<td>Jim A, O, E, B, P, F, R</td>
<td>BS LP, LA, O</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BS O</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>input sentence</th>
<th>active 1st-layer IR units</th>
<th>desired output</th>
<th>actual output</th>
</tr>
</thead>
<tbody>
<tr>
<td>needed break</td>
<td>Jim A, O, E, B, P, F, R</td>
<td>BS PP, AA, O</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BS O</td>
<td></td>
</tr>
</tbody>
</table>
It is interesting to note that the network activated exactly the opposite case units for "Jim" in the sentences. The verb-noun ambiguity of the OBJECT words "fall" and "break" may have also contributed to the error. One simple way to overcome this inconsistency problem is to subdivide each verb type into smaller and more consistent groups. The other way is to regard B and P as the same case roles as proposed in [Cook89].

The third IR sentence that was incorrectly processed corresponds to the following input sentence:

<table>
<thead>
<tr>
<th>input sentence</th>
<th>hammer</th>
<th>sink</th>
</tr>
</thead>
<tbody>
<tr>
<td>active 1st-layer IR units</td>
<td>O, I</td>
<td>LP, O, I, L, S, D</td>
</tr>
<tr>
<td>desired output</td>
<td>O</td>
<td>LP, L</td>
</tr>
<tr>
<td>actual output</td>
<td>O</td>
<td>LP</td>
</tr>
</tbody>
</table>

The lexicalized case unit for LOCATION (L) was not highly activated. This is due to the fact that among the Locative Process verbs, only the verb "sink" has lexicalized case roles. So, the network is heavily biased towards not activating the lexicalized case role. One way to overcome this problem is to subdivide the LP verbs into two groups: one with lexicalized case roles, and the other without.

5.3.2. Generalization and Rule-Based Processing

As discussed previously, the network exhibits remarkable generalization capabilities. After learning to process sentences such as

Jim broke window

the network could correctly process sentences such as

Jim broke glass
Jim broke box
Jim broke sink
Jim broke floor

even though the network did not see these sentences during learning. Note that the IR’s of "glass", "box", and "sink" are different from that of "window", as shown:

| window | O, I, L |
| glass  | O, I   |
| box    | O, I, L, S, D |
| sink   | O, I, L, S, D, LP |
| floor  | O, L, S, D |

The IR’s of "box" and, especially, "sink" and "floor" are significantly different from that of "window". However, since these words share one common property, i.e., their O-bits are all
on, the network can use this information to deduce the correct results. Similar "rule-based" processing occurred when processing other IR sentences. The following are some examples:

\textit{a. Single-IR Generalization}

training sentence \hspace{1cm} \text{window} \hspace{1cm} \text{broke} \\
\hspace{1cm} O,I,L \\

generalize to \hspace{1cm} \text{glass} \hspace{1cm} \text{broke} \\
\hspace{1cm} O,I \\
\hspace{2cm} \text{box} \hspace{1cm} \text{broke} \\
\hspace{2cm} O,I,L,S,D \\
\hspace{3cm} \text{etc.}

\textit{b. Double-IR Generalization}

training sentence \hspace{1cm} \text{hammer} \hspace{1cm} \text{broke} \hspace{1cm} \text{window} \\
\hspace{1cm} O,I \hspace{1cm} O,I,L \\

generalize to \hspace{1cm} \text{window} \hspace{1cm} \text{broke} \hspace{1cm} \text{box} \\
\hspace{1cm} O,I,L \hspace{1cm} O,I,L,S,D \\
\hspace{2cm} \text{box} \hspace{1cm} \text{broke} \hspace{1cm} \text{glass} \\
\hspace{2cm} O,I,L,S,D \hspace{1cm} O,I \\
\hspace{3cm} \text{etc.}

\textit{c. Triple-IR Generalization}

training sentence \hspace{1cm} \text{Jim} \hspace{1cm} \text{moved} \hspace{1cm} \text{glass} \hspace{1cm} \text{from} \hspace{1cm} \text{table} \hspace{1cm} \text{to} \hspace{1cm} \text{floor} \\
\hspace{1cm} O,I \hspace{1cm} O,I,L,S,D \hspace{1cm} O,I,L,S,D \\

generalize to \hspace{1cm} \text{Jim} \hspace{1cm} \text{moved} \hspace{1cm} \text{chicken} \hspace{1cm} \text{from} \hspace{1cm} \text{sink} \hspace{1cm} \text{to} \hspace{1cm} \text{box} \\
\hspace{1cm} O \hspace{1cm} LP,O,I,L,S,D \hspace{1cm} O,I,L,S,D \\
\hspace{2cm} \text{Jim} \hspace{1cm} \text{moved} \hspace{1cm} \text{box} \hspace{1cm} \text{from} \hspace{1cm} \text{floor} \hspace{1cm} \text{to} \hspace{1cm} \text{sink} \\
\hspace{2cm} O,I,L,S,D \hspace{1cm} O,L,S,D \hspace{1cm} LP,O,I,L,S,D \\
\hspace{3cm} \text{etc.}
This kind of "rule-base" processing capability is a direct consequence of our most important design considerations: Constraint Satisfaction, Position Invariance of Influence, and Two-Phase Learning. The use of Two-Phase Learning substantially reduces the overall amount of learning effort required. Instead of having to learn a substantial portion of the 12000 input-output mappings, only 61 word-IR mappings need to be learned in the first phase, and only at most 293 mappings through constraint satisfaction need to be learned in the second phase. The use of Two-Phase Learning, together with Constraint Satisfaction and Position Invariance of Influence, further reduces the required number of IR sentences to 145, or even to a mere 46.

5.3.3. Scaling To Larger Problems

Scaling is less difficult in our network compared to the others because of Two-Phase Learning. In Phase 1, only individual words are involved. Time complexity of this phase is \( O(n f(n)) \) where \( n \) is the number of words and \( f(n) \) is the average amount of time required to learn to classify a word. It is very difficult to estimate \( f(n) \) since it depends on a lot of factors, such as learning rate, network size, etc. The experimental results of Tesravo ([Tesa87]) suggest that \( f(n) \) grows sublinearly with \( n \). So, assuming that \( f(n) \) grows linearly in the worst case, then the time complexity for Phase 1 is only \( O(n^2) \).

Phase 2 involves the learning of rules for mapping IR sentences to desired outputs. Intuitively, if there are \( r \) rules to be learned, Phase 2 will take only \( O(r g(r)) \) amount of time, where \( g(r) \) is the average amount of time required to capture a rule. Now, we need to relate \( r \) to \( n \). Although there may be potentially \( n^w \) input sentences, where \( w \) is the number of words in the longest sentence, \( r \) should be significantly smaller than \( n^w \). The assignment of IRs to words is equivalent to the division of words into groups, each of which has a unique IR. It seems reasonable to suppose that the number of groups required will be more or less a constant, or in the worst case, grows very slowly, say sublinearly. Thus, the number of IRs grows at most in the order of \( n^{1/c} \), \( c > 1 \), and \( r = O(n^{w/c}) \), \( c > 1 \). Experimental results in [PeHa89] suggest that MFT does not spend a lot more time to run than BP, perhaps an increase by only a constant factor. Assuming that \( g(n) \) also grows linearly as for \( f(n) \), then the time complexity for Phase 2 is \( O(n^{w/c + 1}) \), \( c > 1 \). If the compression factor \( c \) is sufficiently large, then time complexity for Phase 2 is just a low-power polynomial. In terms of the actual amount of learning time, it may be further reduced by another factor since our network can generalize to similar rules after learning some of them. Therefore, our network scales better than networks proposed in [McKa86] and [SJMc88] which presumably require learning time of \( O(n^w) \).

5.3.4. Limitations

Our network does have two major limitations. Firstly, our network assumes that input sentences have a fixed maximum length. It is possible to extend the network to accept sentences with indefinite lengths if one can achieve the following:

1. Include a subnetwork, call it STM for short term memory, that stores the results of (local) constraint satisfactions performed on the STM and segments of the input
sentence. The results are stored back into the STM. This seems not too hard to achieve since constraint satisfaction can work on any part of an appropriate network, including the STM and the output units.

2. Prove that a series of local constraint satisfactions is equivalent to global constraint satisfaction performed on the entire sentence at once. Equivalence can be defined in any suitable manner, such as "identical results", "identical costs", and "costs within certain tolerance margin". This is indeed a very difficult problem.

The second limitation is that our network only accepts simple sentences in active voice. Extending it to handle sentences in passive voice and varied tenses seems not too difficult. To extend it to handle complex sentences will be more difficult.

Note that these are also the major limitations of most, if not all, of current neural network solutions for Natural Language Processing tasks.

5.3.5. Advantages Over Other Networks and Symbolic Systems

Here, we summarize the main advantages of our network over the other networks:

1. Our network is able to capture rules better because it models constraint satisfaction, incorporates Two-Phase Learning, and embodies Position Invariance of Influence.

2. It requires less learning effort to achieve good performance. Furthermore, it generalizes remarkably to novel sentences.

3. It scales better to a larger number of sentences. Its performance will not degrade as drastically as the other networks.

The advantages over symbolic systems are:

1. Rules governing case role assignments can be learned by the network. The network can also generalize to (or "infer") other similar rules from those that it has seen.

2. Rules are enforced as soft constraints. Thus, the network is more tolerant to erroneous inputs and is able to compute the best-matched outputs in these situations.

3. The network can be parallelized readily. Although symbolic systems can also be parallelized, their designs still puzzle many researchers.

5.4. Remarks

Two remarks are appropriate in retrospective:
1. The mapping of input words to their IR's is equivalent to a categorization of words. Our categorization of words is quite arbitrary and lacks the linguistic support that the categorization of verbs has. Further research in this area is needed.

2. In our implementation, the 2nd learning phase is modeled as a constraint satisfaction problem. We later found that the 2nd learning phase can also be modeled as a pattern association problem (i.e., replacing the two IR layers by a 3-layered fully-connected feedforward network) with the same level of performance. However, constraint satisfaction network has the advantages that their weights can be initialized appropriately to expedite learning, and they can be interpreted as rules readily.

6. CONCLUSIONS

In this paper, we have discussed a network for performing case analysis, and how we arrive at this network through a careful analysis of the problem. Three most important design considerations are identified, namely Constraint Satisfaction, Two-Phase Learning, and Position Invariance of Influence. Our experimental results confirm that these design considerations are important for designing networks that can achieve good performance with minimal learning effort. The network also generalizes and scales better than other networks.

ACKNOWLEDGEMENT

I like to thank Eric Hartman for providing the MFT program and his help in using the program. His help has been immensely valuable and greatly appreciated.

REFERENCES


APPENDICES

A.1. List of Words

<table>
<thead>
<tr>
<th>verb</th>
<th>verb type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. break</td>
<td>PP, AA</td>
</tr>
<tr>
<td>2. close</td>
<td>PP, AA</td>
</tr>
<tr>
<td>3. eat</td>
<td>AA</td>
</tr>
<tr>
<td>4. enjoy</td>
<td>ES</td>
</tr>
<tr>
<td>5. like</td>
<td>ES</td>
</tr>
<tr>
<td>6. remember</td>
<td>EP</td>
</tr>
<tr>
<td>7. recall</td>
<td>EP</td>
</tr>
<tr>
<td>8. amuse</td>
<td>EP</td>
</tr>
<tr>
<td>9. frighten</td>
<td>EP</td>
</tr>
<tr>
<td>10. taste</td>
<td>EA</td>
</tr>
</tbody>
</table>
11. smell  EA
12. show   EA
13. tell    EA
14. own    BS
15. need   BS
16. owe    BS
17. find   BP
18. lose   BP
19. receive BA
20. borrow BA
21. give   BA
22. lend   BA
23. sit    LS, LA
24. sink   LP
25. fall   LP, LA
26. drop   LP
27. put    LA
28. place  LA
29. move   LA
30. shift  LA

<table>
<thead>
<tr>
<th>noun</th>
<th>case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim</td>
<td>A, O, E, B, P, F, R</td>
</tr>
<tr>
<td>John</td>
<td>A, O, E, B, P, F, R</td>
</tr>
<tr>
<td>Mary</td>
<td>A, O, E, B, P, F, R</td>
</tr>
<tr>
<td>May</td>
<td>A, O, E, B, P, F, R</td>
</tr>
<tr>
<td>he</td>
<td>A, O, E, B, P, F, R</td>
</tr>
<tr>
<td>she</td>
<td>A, O, E, B, P, F, R</td>
</tr>
<tr>
<td>show</td>
<td>O</td>
</tr>
<tr>
<td>movie</td>
<td>O</td>
</tr>
<tr>
<td>fall</td>
<td>O</td>
</tr>
<tr>
<td>break</td>
<td>O</td>
</tr>
<tr>
<td>chicken</td>
<td>O</td>
</tr>
<tr>
<td>beef</td>
<td>O</td>
</tr>
<tr>
<td>money</td>
<td>O</td>
</tr>
<tr>
<td>book</td>
<td>O, I</td>
</tr>
<tr>
<td>glass</td>
<td>O, I</td>
</tr>
<tr>
<td>hammer</td>
<td>O, I</td>
</tr>
<tr>
<td>fork</td>
<td>O, I</td>
</tr>
<tr>
<td>knife</td>
<td>O, I</td>
</tr>
<tr>
<td>window</td>
<td>O, I, L</td>
</tr>
<tr>
<td>door</td>
<td>O, I, L</td>
</tr>
<tr>
<td>box</td>
<td>O, I, L, S, D</td>
</tr>
<tr>
<td>table</td>
<td>O, I, L, S, D</td>
</tr>
</tbody>
</table>
Neural Network for Case Analysis

23. sink    O, I, L, S, D
24. floor   O, L, S, D
25. Austin  L, S, D
26. Dallas  L, S, D
27. Houston L, S, D

prep case marker
1. with MI
2. about MO
3. from MF, MS
4. on ML
5. in ML
6. by ML
7. onto MD
8. to MR, MD

Ambiguous words.

<table>
<thead>
<tr>
<th>verb</th>
<th>when used as noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>as in &quot;tea break&quot;</td>
</tr>
<tr>
<td>show</td>
<td>movie, performance</td>
</tr>
<tr>
<td>fall</td>
<td>water-fall, Autumn</td>
</tr>
<tr>
<td>sink</td>
<td>washing basin</td>
</tr>
</tbody>
</table>

A.2. Case Frames

This is a list of the case frames of the verbs. Verb type SS has no corresponding verbs because these verbs are used in sentences in passive speech.

<table>
<thead>
<tr>
<th>type</th>
<th>case frame</th>
<th>verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>SS</td>
<td>O</td>
</tr>
<tr>
<td>2.</td>
<td>PP</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>PP</td>
<td>I O</td>
</tr>
<tr>
<td></td>
<td></td>
<td>break, close</td>
</tr>
<tr>
<td></td>
<td></td>
<td>break</td>
</tr>
<tr>
<td>3.</td>
<td>AA</td>
<td>A Ox</td>
</tr>
<tr>
<td></td>
<td>AA</td>
<td>A O</td>
</tr>
<tr>
<td></td>
<td></td>
<td>break, close, eat</td>
</tr>
<tr>
<td>4.</td>
<td>ES</td>
<td>E O</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enjoy, like</td>
</tr>
<tr>
<td>5.</td>
<td>EP</td>
<td>E O</td>
</tr>
<tr>
<td></td>
<td></td>
<td>remember, recall</td>
</tr>
<tr>
<td></td>
<td></td>
<td>amuse, frighten</td>
</tr>
<tr>
<td>6.</td>
<td>EA</td>
<td>AE O</td>
</tr>
<tr>
<td></td>
<td>EA</td>
<td>A E O</td>
</tr>
<tr>
<td></td>
<td></td>
<td>taste, smell</td>
</tr>
<tr>
<td></td>
<td></td>
<td>show, tell</td>
</tr>
<tr>
<td>7.</td>
<td>BS</td>
<td>P O</td>
</tr>
<tr>
<td></td>
<td>BS</td>
<td>B O</td>
</tr>
<tr>
<td></td>
<td></td>
<td>own</td>
</tr>
<tr>
<td></td>
<td></td>
<td>need, owe</td>
</tr>
</tbody>
</table>
8. BP BO find, lose
9. BA AR O receive, borrow
   BA AR OF receive, borrow
   BA AR O give, lend
   BA AO R give, lend
10. LS OL sit
11. LP OLx sink
    LP OS fall, drop
    LP OD fall, drop
    LP OSD fall, drop
    LP ODS fall, drop
12. LA AO L sit
    LA AO S fall, move
    LA AO D fall, move
    LA AO SD fall, move
    LA AO DS fall, move
    LA AO L put, place
    LA AO S move, shift
    LA AO D move, shift
    LA AO SD move, shift
    LA AO DS move, shift

A.3. Sample Sentences

Training sentences in Set A are taken from this list after removing duplicates. Symbols in brackets represent the desired outputs.

WINDOW BROKE (O PP)
HAMMER BROKE WINDOW (I PP O)
JIM BROKE WINDOW (A AA O)
JIM BROKE WINDOW WITH HAMMER (A AA O MI I)
WINDOW CLOSED (O PP)
JIM CLOSED WINDOW (A AA O)
JIM ATE CHICKEN (A AA O)
JIM ATE CHICKEN WITH FORK (A AA O MI I)
JIM ENJOYED SHOW (E ES O)
JIM LIKED SHOW (E ES O)
JIM REMEMBERED SHOW (E EP O)
JIM RECALLED SHOW (E EP O)
SHOW AMUSED JIM (O EP E)
SHOW FRIGHTENED JIM (O EP E)
JIM TASTED CHICKEN (AE EA O)
JIM SMELLED CHICKEN (AE EA O)
JIM SHOWED JOHN WINDOW (A EA E O)
JIM TOLD JOHN ABOUT WINDOW (A EA E MO O)
JIM OWNED FALL (P BS O)
JIM NEEDED BREAK (B BS O)
JIM OWED MONEY (B BS O)
JIM FOUND MONEY (B BP O)
JIM LOST MONEY (B BP O)
JIM RECEIVED MONEY (AR BA O)
JIM RECEIVED MONEY FROM JOHN (AR BA O MF F)
JIM BORROWED MONEY (AR BA O)
JIM BORROWED MONEY FROM JOHN (AR BA O MF F)
JIM GAVE JOHN MONEY (A BA R O)
JIM GAVE MONEY TO JOHN (A BA O MR R)
JIM LENT JOHN MONEY (A BA R O)
JIM LENT MONEY TO JOHN (A BA O MR R)
GLASS SAT ON FLOOR (O LS ML L)
GLASS SAT IN SINK (O LS ML L)
JIM SAT ON FLOOR (AO LS ML L)
JIM SAT BY WINDOW (AO LS ML L)
HAMMER SANK (O LPx)
GLASS FELL FROM TABLE (OLP MS S)
GLASS FELL ONTO FLOOR (OLP MD D)
GLASS FELL FROM TABLE ONTO FLOOR (OLP MS S MD D)
GLASS FELL ONTO FLOOR FROM TABLE (OLP MD D MS S)
GLASS DROPPED FROM TABLE (OLP MS S)
GLASS DROPPED ONTO FLOOR (OLP MD D)
GLASS DROPPED FROM TABLE ONTO FLOOR (OLP MS S MD D)
GLASS DROPPED ONTO FLOOR FROM TABLE (OLP MD D MS S)
JIM PUT GLASS ON TABLE (A LA O ML L)
JIM PUT GLASS ON FLOOR (A LA O ML L)
JIM PLACED GLASS ON TABLE (A LA O ML L)
JIM PLACED GLASS ON FLOOR (A LA O ML L)
JIM MOVED FROM HOUSTON (AO LA MS S)
JIM MOVED TO AUSTIN (AO LA MD D)
JIM MOVED FROM HOUSTON TO AUSTIN (AO LA MS S MD D)
JIM MOVED TO AUSTIN FROM HOUSTON (AO LA MD D MS S)
JIM MOVED GLASS FROM TABLE (A LA O MS S)
JIM MOVED GLASS TO TABLE (A LA O MD D)
JIM MOVED GLASS TO FLOOR (A LA O MD D)
JIM MOVED GLASS FROM TABLE TO FLOOR (A LA O MS S MD D)
JIM MOVED GLASS TO FLOOR FROM TABLE (A LA O MD D MS S)
JIM SHIFTED GLASS FROM TABLE (A LA O MS S)
JIM SHIFTED GLASS TO TABLE (A LA O MD D)
JIM SHIFTED GLASS TO FLOOR (A LA O MD D)
JIM SHIFTED GLASS FROM TABLE TO FLOOR (A LA O MS S MD D)
JIM SHIFTED GLASS TO FLOOR FROM TABLE (A LA O MD D MS S)